WIP/-Engineering-Progression

*This is a work-in-progress draft version of a framework for the various engineering levels of the Rail Europe B2C (formerly Loco2) Tech Team. In particular, the salary ranges listed are provided as a **rough guide** and have not been validated by HR, and should not be considered contractual or binding in any way.*

Engineering Progression

Historically, Loco2 primarily hired Senior Software Engineers. As all engineers were at roughly the same level and expected to remain at that level, no advancement was possible or necessary. As the team has grown, we've hired earlier career engineers. Our Senior Engineers have expanded their responsibilities and skill sets. In order to support everyone's career growth, we'd like to introduce a clear progression, indicating responsibilities and required skills for several different levels to support advancement and recognition of exceptional accomplishments.

How to use this framework

Each level is cumulative—for example, Intermediate Software Engineers are expected to possess and demonstrate all of the skills listed for lower levels. Thus, the skills listed will be progressively more brief as you look at each level, for the sake of brevity and to reduce duplication. The levels begin with Software Engineer I (an entry-level position), and currently end with Technical Lead & Principal Software Engineer. These two roles (Technical Lead and Principal Engineer) are at the same level, but have different areas of focus to reflect the specialisation necessary to attain mastery at an advanced level. Skills are divided into four main areas:

- Technical
- Communication
- Delivery
- Leadership

In each area, some of the skills and responsibilities listed may be required; these are indicated with **bold** text. Others are elective (optional). In order to be eligible for promotion to the next level, an engineer should have demonstrated all of the

required competencies at the next level for 3-6 months, as well as at least 50% of optional competencies. Many of the skills listed include detailed examples as italicised sub-points, to make the expectations more explicit.

When considering whether an engineer should be promoted, ask "Does this person demonstrate effective performance for an engineer at the higher level?" If the answer is "yes", a promotion should be granted. If the answer is "no" but the engineer would like to be promoted, their manager should provide a detailed coaching plan to help fill in any competency gaps in time for the next review cycle.

Expected years of experience are provided as a rough guideline, but there is no required minimum. Engineers should be assessed primarily on their competencies and demonstrated skills, not on prior experience. There is no "up or out" expectation that engineers will progress beyond a given level—if you're happy to remain at a Senior Software Engineer level indefinitely, that's perfectly fine. However, we hope that for those engineers who would like to grow their skills in certain areas and to accept greater responsibility, this progression will provide a sufficient framework for advancement with commensurate increase in salary.

Level Summary

Level	Description	Expected Experience	Salary
Software Engineer I	Apprentice	0-1 years (entry level)	£30-35k
Software Engineer II	Junior	1-3 years	35-45k
Intermediate Software Engineer I	Mid-level	2-5 years	40-50k
Intermediate Software Engineer II	Mid-level	3-7 years	45-66k
Senior Software Engineer I	Senior	5+	62-70k
Senior Software Engineer II	Senior	5+	65-80k
Technical Lead	Senior + Technical Leadership, with a focus on mentorship, delivery, value	7+	75-95k (may include bonus)
Principal Software Engineer	Senior + Technical Leadership, with a	7+	75-95k (may include bonus)

	focus on quality, architecture, strategy

Software Engineer I

A Software Engineer writes production-quality code to solve problems and build small enhancements to the new Rail Europe B2C/formerly Loco2 application, with the support and mentorship of more senior team members. A Software Engineer is expected to take a proactive role in their learning progression, reaching out to more senior teammates for mentorship and support when needed to understand how to complete challenging tasks or work through blockers.

Skills

Technical

- **Responsible for maintaining your computer and local development environment**
  - *Keeps local tools up to date*
  - *Follow instructions to get the Rail Europe application running locally*
- **Use version control to manage development workflow**
  - *Follow the dev workflow for creating a pull request and submitting it for review*
  - *Commits are atomic, easy to understand, and don't contain unrelated changes*
  - *You rebase and amend commits to spare your reviewers from sifting through many "fix typo" commits*
- **Use code to fulfil technical tasks (of up to 5 story points, or 1-2 weeks of work) from the prioritised backlog**
  - *Take a ticket from the backlog and makes a necessary change to content or style*
  - *Are assigned a ticket from the backlog and conduct a technical investigation or build a proof-of-concept to illustrate the proposed solution*
- **Maintain appropriate test coverage**
  - *Fix or update tests when changing existing code*
  - *Write a new test to cover a bug fix*
- **Reuse existing code patterns or components**
  - *Follow our guidelines for coding style when adding code or making changes to existing code*
  - *Use idiomatic Ruby when writing new code*

    - *Reuse available React components when working in the SPA*
- **Use continuous delivery or build pipelines for automation**
    - *See your build is failing and finds out why using the Travis or Percy interface*
    - *Restart broken builds after investigating why a failure happened*
- Use observability/monitoring tools (but don't necessarily implement monitoring)
    - *Librato, New Relic, Papertrail, Honeybadger*

Communication

- Review pull requests as a secondary reviewer, and ask questions about what you don't understand
    - *You find a typo in a method name and make a suggestion to fix it*
    - *The work is good but your colleague has neglected to include a useful commit message. You provide positive feedback but request some details that would be helpful to add to the commit message before merging.*
- **Craft pull requests with care and empathy for the reviewer**
    - *PR descriptions are detailed and include screen shots when applicable*
    - *Reviewers can easily understand the context for the change from viewing the PR and code changes, without having to click through multiple links to Trello or related conversations elsewhere*
    - *PR descriptions include detailed manual testing steps*
    - *Irrelevant bits of the PR template have been removed*
- **Accept constructive feedback gracefully and act upon it**
    - *When provided with constructive feedback, you don't respond defensively*
    - *If you disagree with the feedback, ask thoughtful questions until you can come to a shared understanding with the reviewer, then implement requested changes*
    - *Receive feedback that daily stand-ups are not sufficiently detailed, so spend more time reviewing the planned and recent work to include more information in the stand-up*
- **Maintain documentation on the systems you work on, making it easy for future engineers to interact with systems and code**
    - *Update the dev setup instructions to document any problems you encountered in setting up your environment*
    - *Write good commit messages that explain why a change was made*
- Write clear tickets, issues and bug reports that contain the necessary amount of detail to be picked up by other engineers
    - *Add links to the pages that are affected by a bug*
    - *Write steps to reproduce an issue that you've found*
    - *Add screenshots to a ticket to help explain a display bug*

Delivery

- **Maintain focus on the most important task**
  - *Create tickets to capture non-trivial tech debt, rather than getting side-tracked by things not needed to complete the current task*
- **Get well-defined tasks from backlog to production (with support as-needed). Tasks may be up to 1-2 weeks of work.**
  - *Turn a user story into a technical implementation in production*
  - *Raise blockers in timely way*
- Contribute openly to discussions, e.g. on Basecamp
- **Regularly communicate the status of work**
  - *Post a comment on Trello when moving a ticket from one column to another*
  - *Attach pull requests to Trello cards when ready*
- **Ask for help or clarification on tasks when required**
  - *Seek guidance from other engineers, rather than answers*
- **Participate in delivery process**
  - *Move tickets to Testing column when they are deployed and ready for testing*
  - *Write detailed daily stand-ups*

Leadership

- **Act with integrity, honesty and accountability**
- Positively contribute to an inclusive team culture
  - *Tactfully call out exclusive or alienating behaviours from others*
  - *Improve documentation to help new starters*
- Take ownership of your personal development
  - *See some code that you don't understand, and research how it works*
  - *Proactively learn how to use a new tool/language feature*
  - *Read blog posts about technology*
  - *Attend meet-ups or conferences*

Software Engineer II

Software Engineer II indicates an engineering level with some experience, but engineers at this level should be provided with ongoing support and mentorship from more senior team members. Level 2 Software Engineers write production-quality code to solve problems and build enhancements to the new Rail Europe B2C/formerly Loco2 application, and they are expected to exhibit more independence in tackling challenging pieces of work, looking to their colleagues for

guidance and effectively acting upon feedback, whilst contributing to discussions and conversations. Software Engineer II may also begin to specialise in a particular area of the codebase or technological principle.

Skills

Technical

- **Use code to make something (up to 2-3 weeks' work, or 13 story points)**
  - *Take a feature from the prioritised backlog and write the code and tests for that feature*
  - *Breaks work into smaller discrete chunks for easier-to-review pull requests*
- **Ensure your changes are well-tested and don't introduce regressions**
  - *Write automated unit and end-to-end tests for features and bug fixes*
  - *Develop a testing plan and share on the Trello card so product owners and QA know how to test your changes, and in which environment to test them*
  - *Use feature flippers to test on production when possible*
- **Reuse existing code**
  - *Use Ruby `Money` module for currency-related work*
  - *Use an appropriate open-source library when advisable*
- Maintain the security of the systems you work on
  - *Fix vulnerabilities raised by GitHub security alerts*
  - *Conduct investigation into PCI DSS compliance scan results*
- **Regularly and independently debug and fix bugs in your own code**
  - *Fix broken tests caused by changes in their code*
- Get involved in fixing live incidents in production
  - *Responds to alerts for services in production by investigating errors and beginning remedial action*
- Improve our continuous delivery or build pipelines for automation
  - *Make config changes in Travis CI*
  - *Debug problems with Percy configuration or screenshots*
  - *Add or optimise steps to the CI build*
- Use and improve monitoring
  - *Add a new metrics to a space in Librato*
  - *Use Papertrail logs to track down a bug in production*
- **Make pragmatic decisions about technical trade-offs within your own code**
  - *Weighs up the benefits of making code more abstract vs specific*
  - *Reasons about making an API call from the client or from the server*

Communication

- **Create, maintain and enhance documentation for the systems you work on, making it easy for future engineers to interact with systems and code**

    - *Create wiki articles to thoroughly document new features, linking to the relevant Basecamp threads, important Trello cards, and major PRs*
    - *Find some documentation you are reading is out of date so open a Trello card to find the appropriate person to improve it*
- Present your work clearly to a product owner or tech lead
    - *Add an update on Basecamp about a new feature including a video demoing the feature*
    - *Explain your approach to a technical problem to a tech lead*
- **Raise blockers in the appropriate place (to your mentor, in `#tech` or other relevant Slack channel) promptly**

Delivery

- **Lead on getting tasks (up to 13 SPs) from backlog to production**
    - *Turn a well-defined user story into a technical implementation in production*
    - *Investigate a vague bug report and successfully resolve the issue*
- **Effectively collaborate with team members from other disciplines to deliver features**
    - *Work with your product owner to discuss some edge-cases in a feature*
    - *Help to debug a cross-browser issue with a tester*
- Regularly contribute openly to discussions and encourage others to do so too
- **Manage, prioritise and communicate your own workload**
    - *Use Trello, daily stand-ups, 1:1s and other opportunities to communicate about blockers, current status, and get support as-needed*
- Understand our product in a general sense: how it works, what it can do, what features exist

Leadership

- **Know who your project's stakeholders are**
- Share knowledge with peers informally
    - *Come back from a conference and share your learnings with others*
- Influence a community of practice
    - *Answer questions in the `#tech` Slack channel*
    - *Share relevant content/links in Interesting Articles thread on Basecamp*

Intermediate Software Engineer I

An Intermediate Software Engineer writes production-quality code to solve problems and build features in the new Rail Europe B2C/formerly Loco2 application, providing support to more junior team members and reaching out to more senior teammates for guidance as needed. While Intermediate Software Engineers may not know how to approach every task they are assigned, they can independently investigate issues and are self-directed in their learning, able to discover what they don't know and pursue appropriate training and practice to enhance their skills.

Skills

Technical

- **Implement appropriate observability and monitoring when building a solution**
  - *When adding a new dependency to a system, add a healthcheck to monitor the dependency's state*
  - *Add logging that is well-structured and captures useful information about the state of a system*
- **Evaluate third-party software to use in projects**
  - *Can choose between similar Ruby libraries evaluating code quality, ease of integration, future maintenance, and security concerns*
  - *May be involved in evaluating paid-for third party supplier code*
- **Understand the security attack vectors for your area of technology and mitigate against them**
  - *Sanitise user input to mitigate against XSS attacks*
  - *Protect public API endpoints*
  - *Articulate security risks/benefits when evaluating third-party software*
- **Make pragmatic decisions about technical trade-offs within their project**
  - *Know when to stop work on a feature that has fulfilled the requirements vs. spending an extra week on making it perfect but delivering little additional value*
  - *Manage technical debt, understand consequences of technical debt vs the cost of fixing it and act accordingly*
  - *Can explain when something is worth refactoring even when it will impact the speed of delivery*
- **Deliver high-quality code and solutions for tasks up to 1-2 months**
  - *Refactor solutions to improve clarity and maintainability*
- **Regularly and independently debug and fix bugs regardless of origin**
  - *Pick up and debug an urgent issue that comes in to the team, despite having not written the code originally*
- **Choose the appropriate tool, technology or software for a task**
  - *If starting a new project, use tools already understood by the team unless*

there is an agreed good reason to change

Communication

- **Review pull requests and give actionable, empathetic feedback**
  - *PR approvals point out what works well in the code or ask detailed questions about what's not easily understandable*
  - *The work is good but you spot a place where a potential corner case is unhandled. You suggest an enhancement to improve the reliability of the code.*
- **Communicate technical concepts clearly and adapt that communication to the audience**
  - *Explain your work in stand-ups knowing which technical details to leave out to make the message meaningful to everyone in the room*
  - *Teach more junior engineers*
- Facilitate productive discussions with clear outcomes
  - *Run meetings with clear agendas and outcomes*
  - *Obtain wide feedback on technical proposals (e.g., via a Basecamp discussion) and take ownership of seeing it through*
- Contribute to hiring process
  - *Refer engineers for suitable open roles*
  - *Provide constructive feedback about job specs for open roles when they're posted*

Delivery

- Use user research or data to inform decisions
  - *Attend customer-based user research for a feature being worked on*
  - *Set up a testing session with peers for a new bit of tooling*
  - *Find a common pain point among teammates and proposes/builds a solution for it*
- Break down large complex technical proposals into discrete tasks
  - *Create the user stories for an epic or mini-epic with a PO*
- **Communicate work's status upwards to a Tech Lead or manager**
- **Move blockers to enable more junior engineers to work**
  - *Review pull requests in accordance with our guidelines*
  - *Suggest someone to talk to, e.g., "[X] knows the most about [technology Y], you could ask them"*
- Improve delivery process and encourages others to do the same
  - *Champions technical issues that affect delivery such as release cycles, dealing with tech debt and bug fixes*
  - *Encourages other engineers to participate effectively in stand-ups and retrospectives*
- **Build expertise and knowledge in a particular area of our product or**

business

- *Understand how station availability impacts our inventory, and how the stations gem works*
- *Learn to use the various features of our CMS, and support the marketing team in using it*
- *Understand the booking flow and features of the mobile application, including native apps*

Leadership

- Contribute to the personal development of more junior people
  - *Is a designated buddy to a new starter*
  - *Regularly meets up with more junior peers to provide guidance*
  - *Pair with more junior team members*
  - *Write blog posts to share knowledge*

Intermediate Software Engineer II

Intermediate Software Engineer II writes high-quality production code to solve problems and build features in the new Rail Europe B2C/formerly Loco2 application. In addition to possessing the skills expected of an Intermediate Software Engineer I, Intermediate Software Engineer II possesses a sophisticated understanding of their role within the larger organisation and contributes positively to the overall impact of the team. The code they write is secure and performative, and they take ownership of features built to ensure any newly introduced bugs are quickly resolved without impacting their teammates unduly.

Skills

Technical

- **Build products ensuring you take adequate steps to protect sensitive data**
  - *Sensitive data is masked in logs*
  - *Data is retained only for as long as it is needed*
- **Implement appropriate observability and monitoring when building a solution**
  - *Build a Librato dashboard that visualises normal and abnormal operation of a system*
- Lead on fixing live incidents in production
  - *Take proactive action when an incident is reported on a system you support*

and resolves it satisfactorily
    - *Respond to critical issues raised in `#panic` taking the initiative to fix them and report back*
- Encourage others to deliver high-quality code and solutions
    - *Implement tooling to enforce high standards*
    - *Review pull requests fairly & critically in such a way that team members produce better code*
- **Build software or services considering resilience, performance and failure modes**
    - *Combine multiple data sources in a feature, caching, polling, etc., as appropriate to cope with problems in downstream services*
    - *Add healthchecks to a system that detect different ways in which it can fail*
    - *Design and implement a build pipeline*
- **Consider the technical direction of your team or the wider organisation when coming up with technical solutions**
    - *Understand how your work feeds into the organisation's overall tech strategy*
    - *Can articulate and justify the total cost of ownership of your technical solutions*

Communication

- **Communicate technical concepts clearly and adapt that communication to the audience**
    - *Create diagrams to document how the different parts of systems interact*
    - *Present your own work clearly to stakeholders*
- Contribute to hiring process
    - *Participate in hiring panels or technical interviews*
    - *Review tech tests*

Delivery

- **Scopes and prioritises technical work for the team (usually with others)**
    - *You add detailed information and suggested starting points to a task in the backlog so it's ready for a new starter to pick up*
    - *You prioritise work in your area or your project and suggest a few tickets to the technical leads for prioritisation*

Senior Software Engineer I

A Senior Software Engineer writes high-quality production code to solve problems and build features in the new Rail Europe B2C/formerly Loco2 application. The code they write is secure and performative, and they take responsibility for upholding the overall quality of the codebase and the product. They are comfortable with ambiguity, able to work with stakeholders to clarify vague requirements in order to solve problems, and act as a leader and mentor to more junior team members.

Skills

Technical

- **Make pragmatic decisions about technical trade-offs beyond your project**
  - *Can articulate why the overhead of using a third-party system is worth it for your project*
  - *Decide to invest time in building a dashboard for stakeholders to reduce the number of queries they make to the team*
- **Debug and fix complex bugs efficiently**
  - *Investigate a drop in organic traffic from Google, make educated investigations into various aspects of the end-to-end system, consulting other domain experts along the way and keeping stakeholders aware of progress*
  - *Investigate a discrepancy in reported ad traffic. Work with the marketing team to narrow down scope of problem. Use technical knowledge to consult logs for various systems. Identify a fix and implement it.*
- Find technical problems outside of your area and identify ways to improve them
  - *Notice a lot of requests coming in from Customer Support for an admin task that could be automated. Automate the task and work with the Customer Support team on how to use the new tool*
  - *While debugging an issue, trace the bug back to a shared library. Create a patch for the bug and makes sure it is released.*
  - *Spot another team could benefit from using a security feature and help them implement it*
- **Translate difficult business requirements into technical designs**
- Have a deep understanding of, and help others understand, a particular technology or product
  - *Respond to questions on Slack about a particular technology or product*
  - *Provide thoughtful and in-depth feedback on Pull Requests that fall into your area of expertise*
- **Write code that serves as a definitive example for new engineers**
- **Write complex asynchronous and concurrent code**
- **Build maintainable and flexible components and applications**
- Design new schemas comprising multiple tables
- Implement complex asynchronous messaging flows
- **Co-ordinate complex deployments and database migrations**

- Implement distributed systems consisting of multiple interacting services
- **Produce technical designs that include a consideration of scalability**

Communication

- **Present your team's work to others in the business**
  - *Write one-pagers to explain technical decisions to management*
  - *Write a blog post about an aspect of the team's work*
  - *Make sure new features are announced to interested parties in the appropriate Basecamp thread*

Delivery

- **Take a stakeholder problem, investigate to understand it and propose a solution**
- **Tackle complex cross-team technical issues, breaking them down into smaller bits and addressing them**
  - *Manage the roll-out of a new shared tool to multiple code repositories, identifying what work needs to be done, and finding teams to do the work*
  - *Find a bug in a library that affects multiple teams, fixethe bug and work with teams to make sure everybody is able to upgrade*
  - *Find a manual process slowing down multiple teams and automate it*

Leadership

- **Show technical leadership**
  - *Lead on large features or stories*

Senior Software Engineer II

A Senior Software Engineer II writes high-quality production code to solve problems and build features in the new Rail Europe B2C/formerly Loco2 application. They exceed the responsibilities of Senior Software Engineer I by demonstrating leadership and exemplifying our values, demonstrating a high level of involvement and engagement within the organisation, and steering the overall technical strategy in their area of expertise.

Skills

Technical

- **Shape the technical direction for the wider group or tech department**

    - *Contribute to technical strategy work*
    - *Successfully lead the group-wide adoption of a particular technology*
- **Identify and fix security weaknesses**
- **Identify and fix performance bottlenecks in applications**
- Explain all aspects of the web platform to new engineers
- Implement services or libraries that require a deep level of domain knowledge
- **Put users first and can manage competing priorities effectively**
- **Promote accessibility good practice and help other engineers to deepen their accessibility knowledge**
    - *Demonstrate how to use screen readers to other developers*
    - *Improve our documentation with pointers for writing accessible code for other developers*
    - *Take the initiative to report and document accessibility bugs that are currently live in production*

Communication

- **Ask why. Do not take truths for granted unless you understand exactly where they are coming from (especially with regards to regulation, compliance, etc)**
- **Break down delivery and knowledge silos**
- **Keep up-to-date with industry developments and feed specific technical and non-functional recommendations back into the business**
- Proactively identify opportunities to improve company culture around coding standards and non-functional requirements
- Proactively give feedback 'upwards' and to people you interact with who are not in your team

Delivery

- **Break down large projects into smaller iterative steps that each deliver value**
- **Can take a long-term vision (3-4 months) and define building blocks to get there**
- Help Product Managers and Designers to understand and consider non-functional requirements in the product development process

Leadership

- **Show technical leadership**
    - *Shape technical strategy through discussions with the wider organisation*
    - *Act as a technical lead on a project with other developers*
    - *Drive changes to engineering practices with well-reasoned arguments and a*

'strong opinion, weakly held' mentality

- **Share knowledge with others internally**
  - *Give a Lunch & Learn talk*
  - *Add useful tips about development tools and workflows to the wiki or posts on Slack*
  - *More informal knowledge sharing through mentorship*
- **Actively foster an inclusive team culture**
  - *Celebrate good work publicly and encourage the team to do the same*
  - *Spot problems between team members and helps to resolve them*
  - *Models inclusive behaviour to the rest of the team*
- Find learning opportunities for others when reviewing their code and follow it up
  - *I think this code could be improved by doing X, let's pair on it and I'll talk through why X is good for this*

Technical Lead

A Technical Lead leads development across a particular area of the Rail Europe B2C or LocoHub products. They provide technical leadership, coaching, and mentoring to promote knowledge sharing across their team, and take responsibility for the team's overall impact. A Technical Lead's primary contribution may be writing code to solve problems or designing systems to meet business needs, but they will also guide the way in which their team works and uphold a high quality standard. Above all, a Technical Lead has the autonomy to focus on complex problems and create solutions, leveraging the resources of their own expertise as well as that of their team to support business goals.

Skills

Technical

- **Anticipate, identify, and mitigate potential points of failure**
- **Anticipate platform and project needs, technical debt and common issues intuitively**
- **Develop clear technical solutions from ambiguous requirements**
- **Produce technical designs for large complex projects**
- Uncover and fix tricky bugs that have previously evaded detection
- **Demonstrate a deep level of knowledge in a specific area and/or understand the entire architecture for a major part of the business**
- Serve as a technical authority on a technology or an area of the codebase

- Review technical designs and pull requests for large complex projects
- **Encourage and support other engineers to achieve outstanding results**
- Create major contributions to our documentation, and create documents that provide guidelines and best practices to other engineers
- **Work with technical and non-technical stakeholders to translate business requirements into technical designs and implementations**

Communication

- Assist Delivery Manager in hiring process for new Engineers.
  - *Act as an interviewer on an interview panel*
  - *Work to improve the quality of the interviews we conduct and the consistency of the code and CV reviews we do*
- **Help people in non-technical roles understand technical constraints/trade-offs**
- **Can clearly articulate the scaling and reliability limits of your area and escalate awareness of these limits to the management team appropriately**

Delivery

- **Are accountable for the delivery of the team (individually or jointly with other people)**
  - *Work with the delivery manager and product manager to plan upcoming work*
  - *Groom backlog to make sure issues are ready to be picked up*
  - *Proactively unblock others in your team*
- **Help prioritise and balance short-term and long-term investments, focusing on high-impact, high-value work**
  - *Tackle technically challenging work while delegating effectively as-needed*

Leadership

- Help resolve disagreements healthily
  - *Help the team navigate disagreements over the best way to do things. Get agreement and buy-in from engineers on a solution to a problem*
  - *Encourage team members to speak freely in discussions*
  - *Encourage team members to treat each other empathetically*
- **Shape the medium to long-term (6-12 months) priorities of your team**
  - *Find commonalities between small feature ideas in order to form them into larger, coherent technical challenges for the team*
  - *Champion turning things off in order to have capacity to work on new things*
  - *Argue for and form a feature team to tackle a shared problem with other areas of the business*
  - *Write a realistic roadmap in collaboration with a product owner and delivery*

lead

- Make judgements about when to diverge from the immediate goal to achieve something else
- **Responsible for the reliability and maintainability of business-critical systems**
  - *Take ownership of the on-call process, ensuring sufficient out-of-hours coverage in the event of critical errors*
  - *Act as a first-tier responder in the event of a critical incident if the on-call developer cannot be reached*

Principal Software Engineer

A Principal Software Engineer leads development across a particular area of the Rail Europe B2C application or infrastructure. They may do this primarily by writing high-quality production code to solve problems and build features, but they also act as a force multiplier for the rest of their team, using their deep expertise to facilitate growth among their teammates and to maintain an exceptional level of code and product quality. This is a non-managerial technical leadership role, leading by example and shaping the product through strategic advising and direct technical contributions.

Skills

Technical

- **Take responsibility for maintaining standards of excellence in code quality, architecture, test coverage, and software design**
  - *Provide detailed, constructive feedback in code review on pull requests*
  - *Work with other engineers and testers to ensure complex or risky new features are thoroughly tested and safely deployed*
- **Regularly review and triage errors and alerts related to a specific area of the application**
  - *Keep on top of Sentry errors, prioritising maintenance and bug fixes as-needed to keep error rates down*
  - *Notice a common, recurring error and investigates the root cause to resolve it*
- **Maintain awareness of current technologies and frameworks that may support our goals and adopts them as-needed**
  - *Integrate React Hooks and other new framework features into the SPA once they reach maturity*

- Solve the "hard problems" in a given area
- **Find effective technical solutions to larger, ambiguous problems**
- Lead large-scale technical infrastructure projects
- Contribute to external technologies or libraries that we depend on
- **Develop clear technical solutions from ambiguous requirements**
- Uncover and fix tricky bugs that have previously evaded detection
- Implement security improvements that impact multiple services
- Implement performance improvements that impact multiple services
- Demonstrate a deep level of knowledge in a specific area
- **Serve as a technical authority on a technology or an area of the codebase**
- Create dashboards that broadly impact all engineers
- Produce clear technical designs for large complex projects
- **Build systems that serve as definitive examples for new engineers**

Communication

- **Help other people develop themselves and regularly give insightful, useful feedback to those around you**
- Talk to non-technical stakeholders at an appropriate level of abstraction
- **Proactively share knowledge internally**

Delivery

- **Craft proposals for technical solutions to business problems**
  - *Writes a proposal for how we can approach data migration for orders placed on CWS to be accessed on new B2C platform*
  - *Elicits input from the wider tech team and consolidates opinions into a formal recommendation for the management team to review*
- &&Research and identify problems that may appear in the mid-term future**
- Develop strategies and prototypes to mitigate those potential problems

Leadership

- Identify knowledge gaps within the team and gives training to address gaps
  - *Notice that people are not using Git as powerfully as they could so deliver a workshop for engineers on how to use Git's more advanced features.*
  - *Notice you are the only person that understands a particular area of the codebase, so write and deliver a talk at a team meeting about that area.*
- **Work with relevant Engineering Managers to help other engineers perform and grow**
- Foster effective collaboration in multi-disciplinary squads (backend, mobile, data, design, web)
- **Delegate technical decisions with low risk and high reversibility**

- **Own technical decisions with high risk and low reversibility**